VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Modern C++ for Data Structures and Algorithms

Duomenų Struktūros ir Algoritmai Moderniame C++

Kursinis darbas

Author: 4 kurso I grupės studentas
Domas Kalinauskas

Supervisor: Viktoras Golubevas

Vilnius – 2024

Contents

ĮVADAS/INTRO? 3
0.1. Tikslas/Aim 3
0.2. Uždaviniai/Objectives 3
0.3. Ivadas/Intro 3

1. MODERN C++ FEATURES FOR ALGORITHMIC PROBLEM SOLVING 4
1.1. Ranges 4
1.2. Concepts 4
1.3. Functional 4

2. MODERN C++ FOR HIGH-PERFORMANCE COMPUTING 5
2.1. Compile-time calculations 5
2.2. Execution policies 5
2.3. Coroutines 5

3. MEDŽIAGOS DARBO TEMA DĖSTYMO SKYRIAI 6
3.1. Poskyris 6
3.2. Faktorialo algoritmas 6
3.2.1. Punktas 6
3.2.1.1. Papunktis 6
3.2.2. Punktas 6

4. SKYRIUS 7
4.1. Poskyris 7
4.2. Poskyris 7

REZULTATAI IR IŠVADOS 8

IŠVADOS 9

REFERENCES 10

SANTRUMPOS 11

APPENDIXES 12
Appendix 1. Neuroninio tinklo struktūra 12
Appendix 2. Eksperimentinio palyginimo rezultatai 13

Įvadas/Intro?

0.1. Tikslas/Aim

The aim of this <Kursinis darbas> is to analyze how modern C++ improves on the usability of DSA's (data structures and algorithms) and their performance, in comparison to older versions of the C++ standard, and how it stacks up with other languages.

0.2. Uždaviniai/Objectives

1. 2. 3.

0.3. Ivadas/Intro

Collectively, the world is moving towards safer and safer languages. This is seen via widespread adoption of memory-safe languages, such as Rust, or the overall popularity of 'managed', garbage-collected runtimes such as Java, .NET, Python etc. (The safety in 'managed' languages comes from the fact that a vulnerability in the runtime has a much smaller surface area - requiring only a fix in the runtime, instead of the code that is built on top of it)

However, C++ has seen somewhat of a 'renessaince', with a similiar move to safer and more declarative APIs, along with a push towards 'functional' style concepts.

This <Kursinis darbas> aims to be a comprehensive comparison-analysis of recently introduced modern C++ features, and how they stack up against similar features in other languages. We'll <galiu naudot?> take a look at the memory impact, performance impact, and implementation details of these features, along with where they might be ideally used.

Įvade apibūdinamas darbo tikslas, temos aktualumas ir siekiami rezultatai. Darbo įvadas neturi būti dėstymo santrauka. Įvado apimtis 1—2 puslapiai.

1. Modern C++ Features for Algorithmic Problem Solving

1.1. Ranges

A common pattern in C++ code is applying an operation over a selection of elements. The most classic style, which is still in-use today is the index based loop:

```
void operate(std::span<uint32_t> values) {
    for (size_t i = 0; i < values.size(); ++i) {
        std::print("Value: {}", values[i]);
    }
}
```

There are a few notable downsides to the index based for loop, namely that it's error-prone, and can only be effectively used with random-indexable types (e.g. std::array, std::vector). This means that if we want to iterate over a list, or another custom container, we'd have to change the for loop structure to be compatible.

With the introduction of C++11, we got access to the range-based for loop:

```
void operate(const auto &container) {
    for (const auto& x : container) {
        std::print("Value: {}", x);
    }
}
```

When using range-based for loops, we no longer have to manually write the iteration code, meaning it doesn't matter if the type is a array, vector, list, or any other custom iterator. Under the hood, these range-based for loops relies on the container having .begin()/.cbegin() and .end()/.cend() member functions.

1.2. Concepts

1.3. Functional

(Variants, Optional, Expected, Transform, t.t.)

Compare with equivalent features in Rust (mixed-paradigm language) & Haskell (functional language)

2. Modern C++ for High-Performance Computing

2.1. Compile-time calculations

* Compare with other languages that can do that (Zig?) * Mention best compile-time LUT library ¯_(ツ)_/¯ * Maybe something about compression at compile-time

2.2. Execution policies

* Reference TBB * Can be compared to Rust libraries with similiar thing (ehhh forgot but iterator)

2.3. Coroutines

* reference coroutine look-up? (https://www.youtube.com/watch?v=j9tlJAqMV7U) * compare to Rust async & maybe js? async

3. Medžiagos darbo tema dėstymo skyriai

Medžiagos darbo tema dėstymo skyriuose pateikiamos nagrinėjamos temos detalės: pradinė medžiaga, jos analizės ir apdorojimo metodai, sprendimų įgyvendinimas, gautų rezultatų apibendrinimas. Šios dalies turinys labai priklauso nuo darbo temos. Skyriai gali turėti poskyrius ir smulkesnes sudėtines dalis, kaip punktus ir papunkčius.

Medžiaga turi būti dėstoma aiškiai, pateikiant argumentus. Tekste dėstomas trečiuoju asmeniu, t.y. rašoma ne "aš manau", bet „autorius mano", „autoriaus nuomone". Reikėtų vengti informacijos nesuteikiančių frazių, pvz., „...kaip jau buvo minėta...", "...kaip visiems žinoma..." ir pan., vengti grožinės literatūros ar publicistinio stiliaus, gausių metaforų ar panašių meninės išraiškos priemonių.

Skyriai gali turėti poskyrius ir smulkesnes sudėtines dalis, kaip punktus ir papunkčius.

3.1. Poskyris

Citavimo pavyzdžiai: cituojamas vienas šaltinis [PPP01]; cituojami keli šaltiniai [Org00; Pav05a; Pav05b; PPP+02; PPP03; PPŠ04; STU+02; STU01; STU03; STU04; Sur05].

Anglų kalbos terminų pateikimo pavyzdžiai: priklausomybių injekcija (angl. *dependency injection*, dažnai trumpinama kaip *DI*), saitų redaktorius (angl. *linker*).

Išnašų[1] pavyzdžiai[2].

3.2. Faktorialo algoritmas

1 algoritmas parodo, kaip suskaičiuoti skaičiaus faktorialą.

Algorithm 1. Skaičiaus faktorialas

```
1: N ← skaičius, kurio faktorialą skaičiuojame
2: F ← 1
3: for i := 2 to N do
4:     F ← F · i
5: end for
```

3.2.1. Punktas

3.2.1.1. Papunktis

3.2.2. Punktas

[1] Pirma išnaša.

[2] Antra išnaša.

4. Skyrius

4.1. Poskyris

4.2. Poskyris

Rezultatai ir išvados

Rezultatų ir išvadų dalyje turi būti aiškiai išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta) ir pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, teikiamos rekomendacijos, akcentuojamos naujovės).

Išvados

1. Išvadų skyriuje daromi nagrinėtų problemų sprendimo metodų palyginimai, siūlomos rekomendacijos, akcentuojamos naujovės.
2. Išvados pateikiamos sunumeruoto (gali būti hierarchinis) sąrašo pavidalu.
3. Darbo išvados turi atitikti darbo tikslą.

References

[Org00] Organizacijos Pavadinimas. Kodėl abėcėlė vadinasi ABC, o ne DEF? *Žurnalas*. 2000, volume I, pp. 1–20.

[Pav05a] A. Pavardonis. *Bakalauro darbo pavadinimas*. Vilnius, 2005. Bachelor's thesis. Universiteto pavadinimas.

[Pav05b] A. Pavardonis. *Magistrinio darbo pavadinimas*. 2005. Master's thesis. Universiteto pavadinimas.

[PPP+02] A. Pavardenis, B. Pavardonis, C. Pavardauskas, D. Pavardinskas. Straipsnio pavadinimas. In: *Rinkinio pavadinimas*. Miestas, šalis: Leidykla, 2002, pp. 3–15.

[PPP01] A. Pavardenis, B. Pavardonis, C. Pavardauskas. Straipsnio pavadinimas. *Žurnalo pavadinimas*. 2001, volume IV, pp. 8–17.

[PPP03] A. Pavardenis, B. Pavardonis, C. Pavardauskas. *Knygos pavadinimas*. Miestas, šalis: Leidykla, 2003. 172 psl.

[PPŠ04] A. Pavardenis, B. Pavardonis, C. Šavardauskas. *Elektroninės publikacijos pavadinimas*. 2004. [visited on 2015-02-01]. Available from: `https://example.com/kelias/iki/straipsnio`.

[STU+02] A. Surname, B. Tsurname, C. Usurname, D. Vsurname. Article title. In: *Conference book title*. City, country: Publisher, 2002, pp. 3–15.

[STU01] A. Surname, B. Tsurname, C. Usurname. Article Title. *Journal Title*. 2001, volume IV, pp. 3–15.

[STU03] A. Surname, B. Tsurname, C. Usurname. *Book title*. City, country: Publisher, 2003. 172 psl.

[STU04] A. Surname, B. Tsurname, C. Usurname. *Online Source Title*. 2004. [visited on 2015-02-01]. Available from: `https://example.com/path/to/the/article`.

[Sur05] A. Surname. *Title of PhD thesis*. London, 2005. PhD thesis. Title of university.

Santrumpos

Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.

Appendixes

Appendix 1

Neuroninio tinklo struktūra

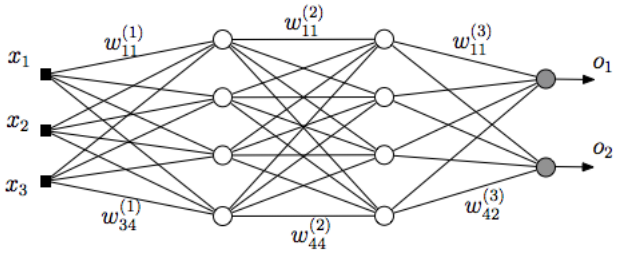

Figure 1. Paveikslėlio pavyzdys

Appendix 2
Eksperimentinio palyginimo rezultatai

Table 1. Lentelės pavyzdys

Algoritmas	$\bar{x}$	σ^2
Algoritmas A	1.6335	0.5584
Algoritmas B	1.7395	0.5647